[11] J. H. Holland, "Genetic algorithms and classifier systems: Foundations and future directions," in *Proc. 2nd Int. Conf. Genetic Algorithms*, J. J. Grefenstette, Ed., 1987.
[12] D. Huang, "A framework for the credit-apportionment process in rule-based systems," *IEEE Trans. Syst. Man Cybern.*, vol. 19, no. 3, pp. 489–498, May/June 1989.
[13] B. Kosko, *Neural Networks and Fuzzy Systems*. Englewood Cliffs, NJ: Prentice Hall, 1992.
[14] D. B. Lenet, "The ubiquity of discovery: Computers and thought lecture," in *Proc. 5th Int. Joint Conf. A.I.*, 1977.
[15] A. D. McAulay, and J. C. Oh, "Image learning classifier system using genetic algorithms," in *Proc. IEEE NAECON '89*, vol. 2/4, pp. 705–710, 1989.
[16] A. D. McAulay, *Optical Computer Architectures*. New York: Wiley, 1991.
[17] J. C. Oh, "Improved classifier system using genetic algorithms applied to image learning," M.S. thesis, Wright State Univ., 1989.
[18] E. Post, "Formal reductions of the general combinatorial problem," *Amer. J. Math.*, vo. 65, pp. 197–268, 1943.
[19] R. L. Riolo, CFS-C: A package of domain independent subroutines for implementing classifier systems in arbitrary, user-defined environments, Tech. Rep., Logic of Computers Group, Univ. Michigan, Ann Arbor, Jan. 1986.
[20] J. D. Schaffer, Some experimenting in machine learning using vector evaluated genetic algorithms, Ph.D. dissertation, Dept. Elec. Eng., Vanderbilt Univ., Nashville, Tennessee, Dec. 1984.
[21] S. F. Smith, A learning system based on genetic algorithms, Ph.D. dissertation, Univ. Pittsburgh, Pittsburgh, PA, 1980.
[22] H. E. Stephanou, and A. P. Sage, "Perspectives on imperfect information processing," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-17, no. 5, pp. 780–789, Sept./Oct. 1987.
[23] S. Watanabe, *Pattern Recognition (Human and Mechanical)*. New York: Wiley, 1985.

A Metric Space Approach to the Specification of the Heuristic Function for the A^* Algorithm

Kevin M. Passino and Panos J. Antsaklis

***Abstract*—Given a graph with arcs that have costs, the A^* algorithm is designed to find the shortest path from a single node to a set of nodes. While the A^* algorithm is well understood, it is somewhat limited in its application due to the fact that it is often difficult to specify the "heuristic function" so that A^* exhibits desirable computational properties. In this paper a metric space approach to the specification of the heuristic function is introduced. It is shown how to specify an admissible and monotone heuristic function for a wide class of problem domains. In addition, when the cost structure for the underlying graph is specified via a metric, it is shown that admissible and monotone heuristic functions are easy to specify and further computational advantages can be obtained. Applications to an optimal parts distribution problem in flexible manufacturing systems and artificial intelligence planning problems are provided.**

Manuscript received March 7, 1991; revised April 10, 1992 and January 22, 1993. This work was supported in part by the Jet Propulsion Laboratory. The work of K. Passino was supported in part by the National Science Foundation under Grant IRI-9210332.

K. M. Passino is with the Department of Electrical Engineering, Ohio State University, Columbus, OH 43210.

P. J. Antsaklis is with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46556.

IEEE Log Number 9212940.

***Index Terms*—A^* Algorithm, artificial intelligence, heuristic search, manufacturing systems.**

I. Introduction

This paper focuses on the computationally efficient solution to an optimization problem on weighted graphs. In particular, we study the problem of how to find the shortest path from a single node to a set of nodes. The graph that is used and the shortest path problem (SPP) considered are defined in Section II. Problems with computational complexity prohibit the use of a conventional dynamic programming solution to the SPP. The standard shortest path algorithms (e.g., Dijkstra's, Moore's, Ford's, and Bellman's algorithms [6]) cannot be used to solve the SPP due to the fact that we search from a node to a *set* of nodes on an *implicit graph* that is *possibly infinite*. It is for these reasons that we utilize a *branch and bound* algorithm called the "A^* algorithm" [9] that can use certain information about the problem domain (to be defined precisely in Section III) to focus the search for a solution to the SPP and, hence, reduce computational complexity. Note that it is possible to solve the SPP via a generalized version of Dijkstra's algorithm but this generalized Dijkstra's algorithm is a special case of A^* [6]. Moreover, in Proposition 2 it is shown that in solving the SPP, if A^* operates with an *admissible* and *monotone heuristic function* it will always visit fewer nodes than the generalized Dijkstra's algorithm.

The problem one encounters, though, is that it is, in general, quite difficult to find an admissible and monotone heuristic function for many applications. Section IV begins by discussing problems with existing results on the specification of the heuristic function. To address these problems we extend the theory of heuristic search by showing that a metric space approach can be used to specify admissible and monotone heuristic functions in a systematic way, for various SPP's, for a wide variety of applications so that they can be solved efficiently. Specifically, the main results of the paper are as follows: Theorem 1 provides a metric space approach to specifying admissible and monotone heuristic functions. Theorem 2 shows that it is not necessary to use a metric to specify the heuristic function. In Theorem 3 and Remark 2 we show how to *automatically* specify admissible and monotone heuristic functions for a wide class of applications that can be modeled via a set of nodes X such that $X \subset \mathbb{R}^n$ (e.g., extended petri nets [35], vector discrete event systems [18], and other Petri net models [12], [16], [38]). In cases where it is known that the costs of the arcs can be specified with a metric, and all the nodes are isolated points, we can expect computational complexity to be further reduced. We introduce a new class of "good" heuristic functions and show in Theorem 4 that these are admissible and monotone. Theorem 6 quantifies how good heuristic functions can be expected to focus the search for solutions to SPP's. The theoretical results in this paper are based upon an extension of those in [26]–[28], [30].

In Section V we apply the method in Section III and results in Section IV to two problems: (1) an optimal part distribution problem in flexible manufacturing systems, and (2) artificial intelligence (AI) planning problems. In each case we show how the results of Section IV can be used to specify admissible and monotone heuristic functions; then we use these in A^* to solve SPP's for both of the applications. The results clearly illustrate that by using our approach to specify the heuristic functions for A^*, significant computational savings can be obtained over the generalized Dijkstra's algorithm for solving SPP's. Concluding remarks are provided in Section VI.

II. The Shortest Path Problem

We consider problems that can be modeled with

$$P = (X, Q, \delta, \chi, x_1, X_f)$$

where

X	is the possibly infinite set of nodes (for our applications in Section V, states),
Q	is the possibly infinite set of labels for arcs (simply called arcs) between nodes (for our applications, these will be "inputs")
$\delta: Q \times X \to X$	is the function (a partial function) which defines a graph,
$\chi: X \times X \to \mathbb{R}^+$	is the *arc cost function* (a partial function)
x_0	is the initial node, and
$X_f \subset X$	is the nonempty finite set of *final nodes*.

$\mathbb{R}^+$ denotes the set of positive reals and $\mathbb{R}_+ = \mathbb{R}^+ \cup \{0\}$. The set

$$E(P) = \{(x, x') \in X \times X: x' = \delta(q, x)\} \cup \{(x_d, x_0)\}$$

denotes the (possible infinite) set of arcs for P (x_d is a dummy node, and (x_d, x_0) a dummy arc added for convenience). The arc cost function $\chi(x, x')$ is defined for all $(x, x') \in E(P)$; it specifies the "cost" for each arc and it is required that there exist a $\delta' > 0$ such that $\chi(x, x') \geq \delta'$ for all $(x, x') \in E(P)$. (For convenience, however, we define $\chi(x_d, x_0) = 0$.) Finally, we require that for each $x \in X$, $|\{\delta(q, x): q \in Q\}|$ is finite, i.e., that the graph of P is *locally finite* (hence, P is equivalent to a δ-graph [10], [34]).

The mathematical notation in this paper is as follows: Let Z be an arbitrary set. Z^* denotes the set of all finite strings over Z including the empty string $\varnothing$. For any $s,t \in Z^*$ such that $s = zz' \cdots z''$ and $t = yy' \cdots y''$, st denotes the concatenation of the strings s and t, and $t \in s$ is used to indicate that t is a substring of s, i.e., $s = zz' \cdots t \cdots z''$. For brevity, the notation $s_{zz''}$ is used to denote a string $s \in Z^*$ such that $s = zz' \cdots z''$ begins with the element $z \in Z$ and ends with $z'' \in Z$. Let z_0 be a distinguished member of the set Z. The notation s_z is used to denote a string $s \in Z^*$ such that $s = z_0 z' \cdots z$ begins with z_0 and ends with $z \in Z$. Furthermore, $s_{z\rangle}$ denotes a string $s \in Z^*$ such that $s = zz'z'' \cdots$ begins with $z \in Z$ and the end element is not specified. The string $s_{\langle z\rangle}$ denotes the string $s \in Z^*$ such that $s = z_0 z' \cdots zz'' \cdots$, i.e., a string that begins at z_0, passes through z, and whose end element is not specified. A (finite directed) *cycle* is a string $s \in Z^*$ such that $s = zz' \cdots z''z$ has the same first and last element $z \in Z$. A string $s \in Z^*$ is *cyclic* if it contains a cycle (for $t_{zz} \in Z^*$, $t_{zz} \in s$), and *acyclic* if it does not. Let $|s|$ for $s \in Z^*$ denote the length of string $s \in Z$, i.e., the number of elements of Z concatenated to obtain s.

A string $s \in X^*$ is called a *path* (for our applications, a state trajectory) of P if for all successive nodes $xx' \in s$, $x' = \delta(q, x)$ for some $q \in Q$. Let

$$E_s(P) \subset E(P)$$

denote the set of all arcs needed to define a particular path $s \in X^*$ that can be generated by P. For some path $s = xx'x''x''' \cdots$, $E_s(P)$ is found by simply forming the pairs (x, x'), (x', x''), x'', $x'''), \cdots$. A sequence of arcs $u \in Q^*$ that produces a path $s \in X^*$ is constructed by concatenating $q \in Q$ such that $x' = \delta(q, x)$ for all $xx' \in s$. Let $X_z \subset X$ then

$$\mathfrak{X}(P, x, X_z) \subset X^*$$

denotes the set of all finite paths $s = xx' \cdots x''$ of P beginning with $x \in X$ and ending with $x'' \in X_z$. Then, for instance, $\mathfrak{X}(P, x_0, X_f)$ denotes the set of all finite length paths for P that begin with the initial node x_0 and end with a final node $x \in X_f$ and $\mathfrak{X}(P, x, X)$ denotes the set of all valid paths for P that begin with $x \in X$. P is said to be (x, X_z)-*reachable* if there exists a sequence $u \in Q^*$ that produces a path $s \in \mathfrak{X}(P, x, X_z)$.

To specify the SPP let the *performance index* be

$$J: X^* \to \mathbb{R}_+$$

where the cost of a path s is defined by

$$J(s) = \sum_{(x,x') \in E_s(P)} \chi(x, x')$$

for all $x \in X$ and $s \in \mathfrak{X}(P, x, X)$. By definition, $J(s) = 0$ if $s = x$ where $x \in X$.

Shortest Path Problem (SPP): Assume that P is (x_0, X_f) reachable. Find a sequence of arcs $u \in Q^*$ that leads P along an *optimal path* s^*, i.e., $s^* \in \mathfrak{X}(P, x_0, X_f)$ such that $J(s^*) = \inf\{J(s): s \in \mathfrak{X}(P, x_0, X_f)\}$.

There may, in general, be more than one optimal path, i.e., the solution to the SPP is not necessarily unique. The set of optimal paths for P, beginning at node $x \in X$, and ending at node $x' \in X_z$, where $X_z \subset X$, is denoted by $\mathfrak{X}^*(P, x, X_z) \subset \mathfrak{X}(P, x, X_z)$. In this paper we are concerned with finding only one optimal path for the SPP and finding it in a computationally efficient manner.

III. Solutions via Heuristic Search

The approach here is to use a *search algorithm* to successively generate candidate paths until an optimal one is found. A brute force approach to solving this problem may produce an algorithm whose computational complexity would prohibit solving all but the simplest of SPP's. Here we use an approach which seeks to minimize the number of paths considered and hence, produces a solution in a computationally efficient number for a wide variety of applications.

A conventional dynamic programming solution could be used for the SPP, but due to the problem of state space explosion, such methods can result in an inefficient algorithm with large memory requirements [1], [40]. Often, a branch and bound technique is chosen in such situations to produce either optimal or near optimal solutions (see, for instance, [6], [15], [17], [22]). This is the approach taken here. We use a particular class of branch and bound algorithms called "heuristic search" algorithms [13], [23] which utilize the "principle of optimality" of dynamic programming and the advantages of branch and bound algorithms that allow certain candidate solutions (paths) to be eliminated from consideration by using information from the problem domain. The particular heuristic search algorithm used here is called the "A^* algorithm" and it was introduced in [4], [9], [10]. The formal properties of A^* are given in [24], [25], [34] and are briefly summarized below to provide the necessary background for this paper.

Note that: (1) $|X|$ can be infinite, (2) the graph of P is defined *implicitly* rather than explicitly, and (3) we search for the shortest path from one node to a *set* of nodes. Hence, Dijkstra's algorithm [6] cannot, in general, be used to solve the above SPP. It is for similar reasons that Moore's, Ford's, and Bellman's algorithms [6] cannot be used to solve the SPP. Dijkstra's algorithm can be generalized so that it can also operate even when (1)–(3) hold; this "generalized Dijkstra's algorithm" is actually a special case of the A^* [6] which will be used here. In fact, below we will show that the worst case computational complexity A^* is always less than or equal to that of the generalized Dijkstra's algorithm. Moreover, for

a wide class of problem domains, we can significantly reduce the amount of computations taken to solve the SPP compared with the generalized Dijkstra's algorithm.

A. Theory of the A* Algorithm

Heuristic search techniques have been applied to search problems where computational complexity is either very high or intractable. The A^* algorithm is one of the most widely used heuristic search algorithms. It utilizes information about how promising it is that particular paths are on an optimal path to reduce the computational complexity. To do this, $J(s^*)$ is estimated by some easily computable *evaluation function* given by $\hat{f}: X^* \to \mathbb{R}_+$ which is defined for all $s \in X^*$ such that $s \in \mathfrak{X}(P, x, X)$ where $x \in X$ (often "$\hat{f}(x)$" is used [34] but we use the mathematically correct notation "$\hat{f}(s_x)$"). If $s^* \in \mathfrak{X}^*(P, x_0, X_f)$ and $s^* = s^*_{\langle x \rangle}$ then $J(s^*) = J(s^*_x) + J(s^*_{xx'})$ where $J(s^*_x) = \min\{J(s_x): s_x \in \mathfrak{X}(P, x_0, \{x\})\}$ and $J(s^*_{xx'}) = \min\{J(s_{xx'}): s_{xx'} \in \mathfrak{X}(P, x, X_f)\}$. The evaluation function $\hat{f}$ is obtained by approximating both $J(s^*_x)$ and $J(s^*_{xx'})$ with appropriately defined functions. The value of $J(s^*_x)$ will be estimated using $\hat{g}: X^* \to \mathbb{R}_+$ where $\hat{g}(s_x) = J(s_x)$ for all $s_x \in \mathfrak{X}(P, x_0, X)$ ("$\hat{g}(x)$" is often used in the literature). Note that $\hat{g}(s_x) = 0$ if $s_x = x_0$ the initial node of P. To estimate $J(s^*_{xx'})$ the function $\hat{h}: X \to \mathbb{R}_+$ is used with $\hat{h}(x) = 0$ if $x \in X_f$. The function $\hat{h}$ is called the "heuristic function" since it provides the facility for supplying the A^* algorithm with special information about the particular search problem under consideration to focus the search of A^*. The evaluation function is chosen to be $\hat{f}(s_x) = \hat{g}(s_x) + \hat{h}(x)$ where $x \in X$. The function $\hat{f}(s_x)$ estimates the cost of a path from x_0 to $x' \in X_f$ that goes through the node x. The A^* algorithm proceeds by generating candidate paths which are characterized with two sets $C \subset E(P)$ and $O \subset E(P)$. The operation of finding the set $\mathcal{E}(x) = \{x' \in X: x' = \delta(q, x)\}$ is called *expanding the node* $x \in X$. For Z and Z' arbitrary sets, let $Z \leftarrow Z'$ denote the *replacement* of Z by Z'. The A^* algorithm which produces an optimal path $s^* \in \mathfrak{X}^*(P, x_0, X_f)$ assuming that P is (x_0, X_f) reachable is given by:

*The * Algorithm*:

(1) Let $C = \{\}$ and $O = \{(x_d, x_0)\}$.

(2) If $|O| > 0$, then go to Step 3. If $|O| = 0$, then exit with no solution.

(3) Choose $(x, x') \in O$ so that $\hat{f}(s_x x')$ is a minimum (resolve ties arbitrarily).
Let $O \leftarrow O - \{(x, x')\}$ and $C \leftarrow C \cup \{(x, x')\}$.

(4) If $x' \in X_f$ then exit with $s_{x'} \in \mathfrak{X}^*(P, x_0, X_f)$, an optimal path.

(5) For each $x'' \in \mathcal{E}(x')$:
(a) If for all $\bar{x} \in X$, $(\bar{x}, x'') \notin C \cup O$ then let $O \leftarrow O \cup \{(x', x'')\}$.
(b) If there exists $\bar{x} \in X$ such that $(\bar{x}, x'') \in O$ and $\hat{f}(s_{x'x''}) < \hat{f}(s_{\bar{x}x''})$ then let $O \leftarrow O - \{(\bar{x}, x'')\}$ and $O \leftarrow O \cup \{(x', x'')\}$.
(c) If there exists $\bar{x} \in X$ such that $(\bar{x}, x'') \in C$ and $\hat{f}(s_{x'x''}) < \hat{f}(s_{\bar{x}x''})$ then let $C \leftarrow C - \{(\bar{x}, x'')\}$ and $O \leftarrow O \cup \{(x', x'')\}$.

(6) Go to step (2).

The contents of C and O change at different stages of the algorithm, but it is always the case that there does not exist $(x^1, x^2) \in C \cup O$ and $(x^3, x^4) \in C \cup O$ such that $x^2 = x^4$ and $x^1 \neq x^3$. Let the set of paths of P, investigated by A^*, be denoted by $\mathfrak{X}(P, C, O)$. Each path $s_{x'} \in \mathfrak{X}(P, C, O)$ begins with x_0, the initial node, and has an end node $x' \in X$ such that $(\cdot, x') \in C \cup O$. For $s, s' \in X^*$ let $s \leftarrow ss'$ denote the operation of *replacing* s by ss'. To find $s_{x'} \in \mathfrak{X}(P, C, O)$ from C and O choose $(x, x') \in C \cup O$ let $s = xx'$. Repeat the following steps until x_d is encountered: (a) find $(x^1, x^2) \in C \cup O$ with $x^2 = x$ where $s = x \cdots$, (b) let $s \leftarrow x^1 s$, and go to (a). The A^* algorithm above is nearly the same as that originally given in [9] except for clarity the "pointers" (arcs) are included explicitly in the algorithm via O and C.

A^* is said to be *complete* since it terminates with a solution. A heuristic function $\hat{h}(x)$ is said to be *admissible* if $0 \leq \hat{h}(x) \leq J(s^*_{\langle x \rangle})$ for all $x \in X$ such that $s^*_{\langle x \rangle} \in \mathfrak{X}^*(P, x, X_f)$. Let $A^*(\hat{h}(x))$ denote an A^* algorithm which uses $\hat{h}(x)$ as its heuristic function. If $\hat{h}(x)$ is admissible then $A^*(\hat{h}(x))$ is said to be *admissible* since it is guaranteed to find an optimal path when one exists, i.e., when P is (x_0, X_f) reachable. A heuristic function $\hat{h}(x)$ is said to be *monotone* if $\hat{h}(x) \leq \chi(x, x') + \hat{h}(x')$ for all $(x, x') \in E(P)$. A heuristic function $\hat{h}(x)$ is said to be *consistent* (equivalent to being monotone) if $\hat{h}(x) \leq J(s^*_{xx'}) + \hat{h}(x')$ for all $(x, x') \in E(P)$ where $s^*_{xx'} \in \mathfrak{X}^*(P, x, x')$. If $\hat{h}(x)$ is a monotone heuristic function then $A^*(\hat{h}(x))$ finds optimal paths to all expanded nodes, i.e., $\hat{g}(s_x) = J(s^*_x)$ for all $x \in X$ with $(\cdot, x) \in C$, $s_x \in \mathfrak{X}(P, C, O)$, and $s^*_x \in \mathfrak{X}^*(P, x_0, x)$. The real utility of knowing that $\hat{h}(x)$ is monotone lies in the fact that nodes are expanded at most once. This implies that the A^* algorithm can be simplified by removing Step 5, part (c) since arcs (pointers) will never be taken from C and placed in O.

B. Efficient Solutions to the Shortest Path Problem

In this Section we show that the A^* algorithm produces efficient solutions to the SPP. The following proposition follows immediately from the above discussion.

Proposition 1: If $\hat{h}(x)$ is admissible then $A^*(\hat{h}(x))$ provides a solution to the SPP.

For a worst case analysis of the complexity of A^* used to solve the SPP in [20] the authors assume as a basic operation the expansion of a node and that $\hat{f}(s_x)$ is easy to compute. Let $X_e = \{x \in X: \hat{f}(s_x) \leq J(s^*), s_x \in \mathfrak{X}(P, C, O), s^* \in \mathfrak{X}^*(P, x_0, X_f)\}$. No more than $|X_e|$ nodes, where $|X_e| \leq |X|$, will be expanded at termination. If $\hat{h}(x)$ is only admissible (and not monotone) then it is possible that A^* expands $O(2^r)$ (where $r = |X_e|$) nodes in the worst case. If $\hat{h}(x)$ is known to be monotone then each node is only expanded once so A^* has complexity $O(|X_e|)$ in the worst case. In general, if it is assumed that visiting a node is the basic operation then if $\hat{h}(x)$ is monotone, A^* runs in $O(|X_e|^2)$ steps in the worst case. (We shall use this latter characterization of computational complexity to compare A^* to other conventional algorithms.) It is also important to note that the computational complexity of A^* is optimized relative to a certain class of algorithms that are "equally informed" about the problem domain and return an optimal solution [2]. In fact, if $\hat{h}(x)$ for A^* is monotone, then A^* uses the most effective scheme of any admissible algorithm for utilizing the heuristic information provided by $\hat{h}(x)$ [2]. The following proposition follows immediately from the above discussions.

Proposition 2: If $\hat{h}(x)$ is monotone and $|X|$ is finite then the complexity of $A^*(\hat{h}(x))$ is $O(|X_e|^2)$ and the complexity of the generalized Dijkstra's algorithm is $O(|X|^2)$ where $|X_e| \leq |X|$.

Proposition 2 indicates that: (1) A^* should always be chosen over the generalized Dijkstra's algorithm to solve the SPP, provided that a monotone $\hat{h}(x)$ can be found, and (2) if a monotone $\hat{h}(x)$ can be found, then $||X| - |X_e||$ or the size of $\hat{h}(x)$ for all $x \in X$, quantifies the computational savings of A^* over the generalized Dijkstra's algorithm. Roughly speaking, the larger that $\hat{h}(x)$ can be chosen (still maintaining monotonicity) the fewer nodes A^* will have to

expand to find an optimal path. Results have in fact shown that for a wide class of graphs, if $\hat{h}(x)$ is monotone then A^* far outperforms the generalized Dijkstra's algorithm. For instance, in the case where $\hat{h}(x)$ is monotone it has been shown that for a wide class of randomly generated "Euclidean graphs" A^* operates with an average complexity of $O(|X|)$ [37]. Similar results on the reduction of search complexity obtained with A^* over the generalized Dijkstra's algorithm are provided in [5].

IV. The Heuristic Function

It is clear that it is very important to be able to specify an $\hat{h}(x)$ that is monotone; otherwise the complexity of A^* can become exponential in the worst case. Unfortunately it is not easy to specify monotone heuristic functions for a wide class of applications; hence, the use of A^* has been somewhat limited to special situations. This problem is partly resolved here by showing that a metric space approach provides a method to specify monotone (and hence admissible) heuristic functions for a very wide class of applications.

There has been extensive work on the problem of how to automatically generate heuristics for an arbitrary problem. In [3], [7], [8], [33] the authors introduced, respectively, the related "problem similarity," "auxiliary problem," and "relaxed model" approaches to the generation of heuristics. The main deficiencies of these approaches is that they provided no way to systematically produce similar and auxiliary problems or relaxed models. Furthermore, in [39] it was proven that the approach in [7] can be computationally inefficient. Approaches similar to these have also been used in Operations Research [11], [17]. As an extension to Pearl's (and the others) work, the authors in [14] suggest a method for modeling a problem that will always lead to the derivation of a set of "simplified" subproblems from which admissible and monotone heuristics can be derived algorithmically for the original problem. Their algorithm uses a problem decomposition algorithm to obtain the subproblems and then uses exhaustive search to find the minimal cost optimal path in each subproblem. From this, a heuristic which is admissible and monotone is generated. The problem with this approach is the reliance on an exhaustive search. While Irani and Yoo have found computationally efficient solutions to several specific simple problems, the approach of decomposing the problem to generate heuristics was not proven to be computationally efficient in general. Recently, it has been shown that for a class of "vector discrete event systems," a linear integer programming approach can be used to specify the heuristic function for A^* [19]. Unfortunately, computationally efficient techniques do not currently exist to solve the linear integer programming problem.

A. Specifying The Heuristic Function: A Metric Space Approach

In our metric space approach to specifying the heuristic function there is no need to perform a search or use a mechanical decomposition procedure to find the heuristic. In this way we do not defeat the main purpose of using the A^* algorithm-to reduce the computational complexity of search. We will, however, require for some of the results below that P has nodes that are "numerical," i.e., that $X \subset \mathbb{R}^n$. In this way we exploit the structure of X to obtain efficient solutions to the SPP.

Let Z be an arbitrary nonempty set and let ρ: $ZxZ \rightarrow \mathbb{R}$ where ρ has the following properties: (1) $\rho(x, y) \geq 0$ for all $x, y \in Z$ and $\rho(x, y) = 0$ iff $x = y$, (2) $\rho(x, y) = \rho(y, x)$ for all $x, y \in Z$, and (3) $\rho(x, y) \geq \rho(x, z) + \rho(z, y)$ for all $x, y, z \in Z$ (triangle inequality). The function ρ is called a *metric* on Z and $\{Z; \rho\}$ is a *metric space*. Let $z \in Z$ and define $d(z, Z) = \inf\{\rho(z, z'): z' \in Z\}$. The value of $d(z, Z)$ is called the *distance between point z and set Z*. Recall that if $x, y \in \mathbb{R}^n$, $x = [x_1 \ x_2 \cdots x_n]'$, $y = [y_1 \ y_2 \cdots y_n]'$, and $1 \leq p \leq \infty$, then $\rho_p(x, y) = [\sum_{i=1}^{n} |x_i - y_i|^p]^{1/p}$, $\rho_\infty(x, y) = \max\{|x_i - y_1|, |x_2 - y_2|, \cdots, |x_n - y_n|\}$, and ρ_d (*discrete metric*) where $\rho_d(x, y) = 0$ if $x = y$ and $\rho_d(x, y) = 1$ if $x \neq y$, are all valid metrics on $\mathbb{R}^n$ [21]. We shall frequently use these metrics in the following results and in Section V.

The first theorem says that if the heuristic function is chosen to be the distance between a node x and a set X_f as defined in a metric space, and the metric satisfies a certain constraint, then it will be both admissible and monotone.

Theorem 1: For P if $\hat{h}(x) = \inf\{\rho(x, x_f): x_f \in X_f\}$ and ρ is a metric on X with $\rho(x, x') \leq \chi(x, x')$ for all $(x, x') \in E(P)$ then $\hat{h}(x)$ is admissible and monotone.

Proof: For admissibility let $s_{\bar{x}}x'' \in \mathfrak{X}(P, \bar{x}, X_f)$ where $\bar{x} \in X$ and let $xx' \in s_{\bar{x}}x''$ be two successive nodes on $s_{\bar{x}}x''$. From the triangle inequality, $\rho(x, x'') \leq \rho(x, x') + \rho(x', x'')$. Using repeated applications of the triangle inequality along $s_{\bar{x}}x''$ we know that if $t = s_{\bar{x}}x''$

$$\rho(\bar{x}, x'') \leq \sum_{(x,x') \in E_t(P)} \rho(x, x')$$

and with the assumption that $\rho(x, x') \leq \chi(x, x')$ for all $(x, x') \in E(P)$

$$\sum_{(x,x') \in E_t(P)} \rho(x, x') \leq \sum_{(x,x') \in E_t(P)} \chi(x, x').$$

Since this is true for any path it is true for optimal ones also. Let $s^*_{\bar{x}x''} \in \chi^*(\rho, \bar{x}, X_f)$ (we need only consider cases where one exists). Then, from above,

$$0 \leq \rho(\bar{x}, x'') \leq \sum_{(x,x') \in E_t(P)} \chi(x, x') = J(s^*_{\bar{x}x''})$$

where $t = s^*_{\bar{x}x''}$. So, by the definition of $\hat{h}(x)$ we have $0 \leq \hat{h}(\bar{x}) \leq J(s^*_{\bar{x}x''})$ for all $\bar{x} \in X$ and $s^*_{\bar{x}x''} \in \mathfrak{X}^*(P, \bar{x}, X_f)$ which guarantees the admissibility of $\hat{h}(x)$. For monotonicity, let $s_{xx''} \in \mathfrak{X}(P, \bar{x}, X_f)$ where $\bar{x} \in X$ and let $xx' \in s_{\bar{x}x''}$ be two successive nodes on $s_{\bar{x}x''}$. Notice that for the sequence of nodes $x \in X$ expanded, the node at which the inf is achieved in $\hat{h}(x) = \inf\{\rho(x, x_f): x_f \in X_f\}$ may change. Let x_p denote the node at which the inf is achieved for x and x'_p the one for x'. By the triangle inequality, $\rho(x, x'_p) \leq \rho(x, x') + \rho(x', x'_p)$. But by definition of $\hat{h}(x)$ we know that $\rho(x, x_p) \leq \rho(x, x'_p)$. It follows that $\rho(x, x_p) \leq \rho(x, x') + \rho(x', x'_p)$. By the definition of $\hat{h}(x)$ we have $\hat{h}(x) \leq \rho(x, x') + \hat{h}(x')$ and since $\rho(x, x') \leq \chi(x, x')$, $\hat{h}(x) \leq \chi(x, x') + \hat{h}(x')$ for all $x, x' \in X$ such that $xx' \in s$ where $s \in \mathfrak{X}(P, \bar{x}, X_f)$ which guarantees the monotonicity of $\hat{h}(x)$ (we could have just proven monotonicity since it implies admissibility). ■

Remark 1: Assume that P is (x_0, X_f) reachable. Consider the following suboptimal shortest paths problem (SOSPP): Find a sequence of arcs $u \in Q^*$ that leads P along a *near-optimal* (ϵ-*optimal*) *path* s, i.e., $s \in \mathfrak{X}(P, x_0, X_f)$ such that $J(s) \leq (1 + \epsilon)J(s^*)$ where $J(s^*) = \inf\{J(s): s \in \mathfrak{X}(P, x_0, X_f)\}$ and $\epsilon \geq 0$. A solution to this SOSPP is provided in [29] where if ϵ is very small and $\hat{h}(x)$ is ϵ-monotone $\hat{h}(x) \leq (1 + \epsilon)\chi(x, x') + \hat{h}(x')$ for all $(x, x') \in E(P)$) the complexity of the algorithm may be satisfactory for special problems (but it is exponential in the worst case). In [29] the au-

thors use the same metric space approach as above to provide the first results on the automatic specification of $\hat{h}(x)$ for a SOSPP's (first results for automatic specification of "semiadmissible" heuristics [34]). The result is the same as for Theorem 1 except it is required that $\rho(x, x') \leq (1 + \epsilon)\chi(x, x')$ for all $(x, x') \in E(P)$ to get ϵ-monotonicity and hence ϵ-optimality.

It is, however, not necessary to use the metric space notion of distance for the heuristic function as Theorem 2 shows.

Theorem 2: Let $\theta: X \times X \to \mathbb{R}_+$ and suppose that $\theta(x, x') \leq \chi(x, x')$ for all $(x, x') \in E(P)$. For P there exist heuristic functions $\hat{h}(x) = \inf\{\theta(x, x'): x' \in X_f\}$ such that θ is not a metric on X, that are admissible and monotone.

Proof: Suppose that $\theta(x, x') = 0$ for all $x, x' \in X$. Then θ is not a metric but when θ is used in the heuristic function we have $\hat{h}(x) = 0$ for all $x \in X$ which is clearly an admissible and monotone heuristic function. Also, if $\hat{h}(x) \leq \hat{h}'(x)$ for all $x \in X$, where $\hat{h}'(x)$ satisfies the conditions of Theorem 4, $\hat{h}(x)$ is admissible but not necessarily monotone. ■

Theorems 1 and 2 place the statements made in the theory of heuristic search about "distance" between points, and between points and sets in a precise mathematical setting. Likewise, they clarify the relationship between monotonicity and the triangle inequality which has, only in the past, been loosely referred to [9], [34].

Theorem 1 can make it easier to specify $\hat{h}(x)$ because for many problems the conditions of Theorem 1 are easier to test than the admissibility and monotonicity conditions. Theorem 1 does not, however, make the task of specifying $\hat{h}(x)$ entirely simple; $\hat{h}(x)$ still must be chosen so that the constraint $\rho(x, x') \leq \chi(x, x')$ is met for all $(x, x') \in E(P)$ and one must, in fact, be able to specify a metric ρ on X. Theorem 3 and the following discussions show several ways to overcome these difficulties.

Let ρ be any metric on X and

$$\rho_a(x, x') = \beta \frac{\rho(x, x')}{1 + \rho(x, x')}$$

where $\beta = \inf\{\chi(x, x'): (x, x') \in E(P)\}$. Let ρ_b be a bounded metric on X for $(x, x') \in E(P)$, i.e., for all $(x, x') \in E(P)$ there exists $\sigma > 0$ such that $\rho_b(x, x') \leq \sigma$. Let ρ_c be a metric on X and assume that $\rho_c(x, x') = \gamma\chi(x, x')$ for all $(x, x') \in E(P)$ for some $\gamma > 0$. Let ρ_β be a metric on X such that $\rho_\beta(x, x') = \beta$ if $x \neq x'$ and $\rho_\beta(x, x') = 0$ if $x = x'$ for all $(x, x') \in E(P)$.

Theorem 3: For P the heuristic functions:

$$\begin{aligned}
&(1)\ \hat{h}_1(x) = \inf\{\rho_a(x, x_f): x_f \in X_f\}\\
&(2)\ \hat{h}_2(x) = \inf\{(\beta/\sigma)\rho_b(x, x_f): x_f \in X_f\}\\
&(3)\ \hat{h}_3(x) = \inf\{(1/\gamma)\rho_c(x, x_f): x_f \in X_f\}\\
&(4)\ \hat{h}_4(x) = \inf\{\rho_\beta(x, x_f): x_f \in X_f\}
\end{aligned}$$

are all admissible and monotone.

Proof: Due to the fact that there exists $\delta' > 0$ such that $\chi(x, x') \geq \delta'$ for all $(x, x') \in E(P)$, $\beta > 0$; hence, it is easy to show that ρ_a, $(\beta/\sigma)\rho_b$, $(1/\gamma)\rho_c$, and ρ_β are all metrics on X. For (1), $\rho_a(x, x') \leq \chi(x, x')$ for all $(x, x') \in E(P)$ since $\rho_a(x, x') \leq \beta$ and $\beta \leq \chi(x, x')$ for all $(x, x') \in E(P)$. For (2), since $1 \leq (1/\beta)\chi(x, x')$ and $(1/\sigma)\rho_b(x, x') \leq 1$ for all $(x, x') \in E(P)$ we know that $(\beta/\sigma)\rho_b(x, x') \leq \chi(x, x')$ for all $(x, x') \in E(P)$. Clearly for (3), $(1/\gamma)\rho_c(x, x') \leq \chi(x, x')$ and for (4) $\rho_\beta(x, x') \leq \beta$ for all $(x, x') \in E(P)$. With this, the result follows immediately from Theorem 1. ■

To choose $\hat{h}_1(x)$ determine β and specify any metric ρ on X; using ρ_a and Theorem 3, $\hat{h}_1(x)$ will be admissible and monotone. To choose $\hat{h}_2(x)$ pick a bounded metric ρ_b on X and determine σ; using Theorem 3, $\hat{h}_2(x)$ will be admissible and monotone. For $\hat{h}_3(x)$ it must be the case that the costs are in a special form then $\hat{h}_3(x)$ will be admissible and monotone. For $\hat{h}_4(x)$, ρ_β can be specified for any X; hence, this choice will always be admissible and monotone (See Theorem 5 below also).

Notice that for each of the techniques, in order to specify $\hat{h}(x)$ it is necessary to be able to specify a metric on X. In general, this may not be an easy task but as the next Remark and the following comments explain, there are a wide variety of applications for which it is easy to specify a metric on X.

Remark 2: There is a wide class of applications whose graph can be modeled in terms of $X \subset \mathbb{R}^n$, e.g., X comprised of n-tuples of natural numbers or integers. As evidence of this fact we turn to the many applications of the theory of Petri nets [35] (e.g., general or extended petri nets), the use of such models in discrete event system theoretic research [12], [16], [38], [36], and other related "vector discrete event system models" [18].

Theorem 3 says that there is no difficulty in specifying $\hat{h}(x)$ for all problems that can be modeled with P provided a valid metric ρ on X can be specified. Remark 2 indicates that there exists many problems that can be modeled as having a state space $X \subset \mathbb{R}^n$; hence, there is no trouble specifying an admissible and monotone heuristic for the wide class of applications with $X \subset \mathbb{R}^n$ because there exist many metrics on $\mathbb{R}^n$ (e.g., ρ_p, ρ_d, and ρ_∞) and any metric on $\mathbb{R}^n$ is also a metric on X, where $X \subset \mathbb{R}^n$. Note that for particular applications many results similar to Theorem 3 exist, since for ρ_p and ρ_∞ one can weight the various terms in the sum and max respectively; hence, one has flexibility in specifying the heuristic function when this metric space approach is used.

B. Good Heuristic Functions

Theorems 1 and 3 provide an *automatic* procedure to specify $\hat{h}(x)$ for a wide class of problems; the use of such $\hat{h}(x)$ will allow A^* to produce solutions to SPP's in a computationally efficient manner. Next, we seek to show how to make $\hat{h}(x)$ large so that even more computational savings can be obtained, i.e., fewer nodes will be expanded in finding the optimal path.

Consider $P' = (X, Q, \delta, \chi', x_0, X_f)$ defined as for P except χ': $X \times X \to \mathbb{R}_+$ where χ' is a metric on X, i.e., the costs for the arcs are characterized by a metric. Also, in terms of the metric space $\{X, \chi'\}$ every $x \in X$ is assumed to be an isolated point. Notice that, in general, we are requiring that χ' be defined on some (x, x') such that $(x, x') \notin E(P)$. We call a heuristic function *good* if $\hat{h}(x) = \inf\{\chi'(x, x_f): x_f \in X_f\}$ for all $x \in X$. The motivation for our definition of this new class of heuristic functions lies in the desire to choose $\hat{h}(x)$ as large as possible to get efficient search.

Theorem 4: For P' if $\hat{h}(x)$ is good then $\hat{h}(x)$ is admissible and monotone.

Proof: Since every $x \in X$ is an isolated point there exists a $\delta' > 0$ such that $\chi'(x, y) \geq \delta'$ for every $x, y \in X$ such that $x \neq y$. Since A^* prunes cycles it will not repeatedly investigate any single $(x, x') \in E(P')$ with $x = x'$ and $\chi'(x, x') = 0$; hence, if $\hat{h}(x)$ is good then $A^*(\hat{h}(x))$ is complete. By Theorem 1, $\hat{h}(x)$ is admissible and monotone. ■

This indicates that if we have a problem domain P' without costs for the arcs or a problem domain where it is not known how to specify the costs, then Theorem 4 offers a method to assign the costs so that an efficient search for a solution to several SPP's is possible. In fact, for any P such that all the costs are equal (or where this can be assumed) the following result provides an admissible and monotone heuristic function.

Theorem 5: If $\chi'(x, x') = \gamma\rho_d(x, x')$ for all $(x, x') \in E(P)$ for some $\gamma > 0$ then $\hat{h}(x) = \inf\{\chi'(x, x_f): x_f \in X_f\}$ is an admissible and monotone heuristic function for finding the solution to the SPP in the case where the costs are all equal to some γ where $\gamma > 0$.

Proof: Even though $\chi'(x, x') = 0$ when $x = x'$, such self-loops will be pruned by A^* so it does not matter that χ' doesn't precisely model the fact that all the costs are equal. The $(x, x') \in E(P)$ such that $\chi(x, x') \neq \gamma$ cannot be on any optimal path. The result follows directly from Theorem 4 since χ' is a metric. ■

Theorem 5 is quite useful in practice since often a solution is sought which will minimize the length of the path. Theorems 4 and 5 illustrate how information from the problem domain (the knowledge that the costs were modeled with a metric) is used to focus A^*'s search for an optimal solution. This is further quantified by showing that if a good heuristic function $\hat{h}(x)$ is used we can expect $A^*(\hat{h}(x))$ to more narrowly focus its search.

Theorem 6: For P' if $\hat{h}(x) = \inf\{\chi'(x, x_f): x_f \in X_f\}$ for all $x \in X$ then $|\hat{h}(x) - \hat{h}(x')| \leq \chi'(x, x')$ for all $(x, x') \in E(P')$.

Proof: From monotonicity $\hat{h}(x) \leq \chi'(x, x') + \hat{h}(x')$ for all $(x, x') \in E(P')$. Also, with a simple rearrangement, $-\chi'(x', x) \leq \hat{h}(x) - \hat{h}(x') \leq \chi'(x, x')$. Since χ' is a metric, $\chi'(x, x') = \chi(x', x)$ for all $x, x' \in X$ so we have $|\hat{h}(x) - \hat{h}(x')| \leq \chi'(x, x')$ for all $(x, x') \in E(P')$. ■

We see that if the heuristic function is monotone then the estimate of the remaining cost at the next node cannot be too much *smaller* than the estimate of the remaining cost at the current node. This tends to guarantee that we have good heuristic information (large $\hat{h}(x)$) so fewer nodes will be expanded. If χ' is a metric which specifies the costs for the arcs and is used to guide the search, then it is also the case that the estimate of the remaining cost at the next node cannot be too much *larger* than the estimate of the remaining cost at the current node. This tends to guarantee that A^* will not get sidetracked too much from finding an optimal solution.

Theorems 4–6 support the results in [37] where the authors show that if the costs can be defined by a metric, then A^* has average complexity $O(|X|)$ for a wide class of randomly generated graphs and thus, on the average, far outperforms conventional algorithms in solving the SPP. We see that when the heuristic function is based on a metric that is used to specify the costs of the arcs for P', then enough information from the problem domain is used so that we are guaranteed to get an admissible and monotone heuristic function. Hence, SPP's can be solved efficiently.

V. Applications

In this Section we apply the method in Section III and results in Section IV to two problems (other applications are given in [29], [30], [32]): (1) an optimal part distribution problem in flexible manufacturing systems, and (2) artificial intelligence (AI) planning problems. In each case, we specify the model P for the problem and state the particular SPP. Then, using the results of Section IV we specify admissible and monotone heuristic functions so that A^* can find solutions to the SPP's in a computationally efficient manner. A^* and the generalized Dijkstra's algorithm were implemented to compare the complexity of the two algorithms. For all cases in the examples, A^*, using a heuristic function chosen via the results in Section IV, significantly outperformed the generalized Dijkstra's algorithm.

A. Optimal Parts Distribution Problem in Flexible Manufacturing Systems

A flexible manufacturing system (FMS) that is composed of a set of identical machines connected by a transportation system is described by a directed graph (M, T) where $M = \{1, 2, \cdots, N\}$ represents a set of machines numbered with $i \in M$ and $T \subset M \times M$ is the set of transportation tracks between the machines. We assume that (M, T) is *strongly connected*, i.e., that for any $i \in M$ there exists a path from i to every other $j \in M$. This ensures that no machine is isolated from any other machine in the FMS. Each machine has a queue which holds parts that can be processed by any machine in the system (with proper setup). Let the number of parts in the queue of machine $i \in M$ be given by $x_i \geq 0$. There is a robotic transporter that travels on the tracks represented by $(i, j) \in T$ and moves parts between the queues of various machines. The robot can transfer parts from any $i \in M$ to any other $j \in M$ on any path between i and j (it is assumed that the robot knows the path to take, if not A^* could be used to find it). The robot can transfer no more than $\beta \in \mathbb{N} - \{0\}$ parts at one time between two machines. It is assumed that the robot knows the initial distribution of parts and the graph (M, T). We wish to find the sequence of inputs to the robot of the form "move α ($\alpha \leq \beta$) parts from machine i to machine j" that will achieve an even distribution of parts in the FMS. In this way, we ensure that every machine in the FMS is fully utilized. It is assumed that no new parts arrive from outside the FMS and that no parts are processed by the machines while the redistribution takes place. Our example is similar to the "load balancing problem" in Computer Science except that we require that a minimum number of parts be moved to achieve an even distribution. Next, we specify the model P of this FMS.

Let $X = \mathbb{N}^N$ denote the set of nodes (actually, states) and $\boldsymbol{x}_k = [x_1 \ x_2 \ \cdots \ x_N]^t$ and $\boldsymbol{x}_k + 1 = [x_1' \ x_2' \ \cdots \ x_N']^t$ denote the current and next state, respectively. Let $Q = \{u_{ij}^{\alpha}: \alpha \in \mathbb{N} - \{0\}\}$ be the set of arcs (actually, inputs) where u_{ij}^{α} denotes the command to the robot to move α parts from machine i to machine j. The state transition function is given by $\delta(u_{ij}^{\alpha}, \boldsymbol{x}_k) = [x_1 \ x_2 \ \cdots \ x_i - \alpha \ \cdots \ x_j + \alpha \ \cdots \ x_N]^t$, the arc cost function by $\chi(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}) = \alpha$, and $\boldsymbol{x}_0 = [x_{01} \ x_{02} \ \cdots \ x_{0N}]^t$. The set X_f characterizes the state (or states) for which we consider the parts in the FMS to be at an even distribution. Let int (x) denote the integer part of x (e.g., int(3.14) = 3) and "mod" denote modulo. Let

$$L = \text{int}\left(\sum_{i=1}^{N} \frac{x_{0i}}{N}\right) \quad \text{and} \quad L_e = \left(\sum_{i=1}^{N} \frac{x_{0i}}{N}\right) \bmod N.$$

The value of L represents the amount of parts each machine would have if the parts could be evenly distributed and L_e represents the number of extra parts that we seek to distribute across the first L_e machines. With this intent we let $\boldsymbol{x} = [\bar{x}_1 \ \bar{x}_2 \ \cdots \ \bar{x}_N]^t$ where $\bar{x}_i = L + 1$ for i, $i \leq L_e$ and $\bar{x}_j = L$ for j, $L_e < j \leq N$ (other states where the parts are considered to be evenly distributed can be specified in a similar manner—an example of this is given below). We often let $X_f = \{\bar{\boldsymbol{x}}\}$, hence $\hat{f}(s_x)$ is easy to compute. Also note that for each $\boldsymbol{x} \in X$ there are at most $\beta(N - 1)N$ next states which will clearly be much less than $|X|$.

The SPP for this optimal parts distribution problem involves finding a sequence of inputs u_{ij}^{α} to the robot which will result in it moving the least number of parts to achieve an event distribution, i.e., $x_k \in X_f$. By Proposition 1, if we can find an $\hat{h}(x_k)$ that is admissible, then A^* will solve the SPP (possibly inefficiently). Here, we show that the metric space approach developed in Section IV can be used to specify a monotone $\hat{h}(x_k)$ (and hence admissible) so that the SPP can be solved efficiently. First, consider using the metric ρ_p with $p = 1$. Notice that $\rho_1(x_k, x_{k+1}) = 2\alpha$ for all $(x_k, x_{k+1}) \in E(P)$. Hence, by Theorems 3 and 1, $\hat{h}_1(x_k) = (1/2)\rho_1(x_k, \bar{x}$ ($\bar{x}$ defined above) is admissible and monotone so we get an efficient solution to the SPP. Theorems 4 and 6 offer another possibility. Consider the metric ρ_∞. Notice that $\rho_\infty(x_k, x_{k+1}) = \alpha$ for all $(x_k, x_{k+1}) \in E(P)$, all $x_k \in X$ are isolated points, and hence $\hat{h}_\infty(x_k) = \rho_\infty(x_k, \bar{x})$ ($\bar{x}$ defined above) is a good heuristic function. By Theorem 4 it is admissible and monotone.

Consider the FMS with 3, 4, and 6 machines and track topologies shown in Fig. 1. For the 3-machine FMS in Fig. 1 let $\beta = 1$ and $x_0 = [10\ 0\ 4]^t$; then $L = 4$ and $L_e = 2$ and we choose $X_f = \{[5\ 5\ 4]^t\}$. $A^*(\hat{h}_1(x_k))$ and $A^*(\hat{h}_\infty(x_k))$ both expand 5 states and result in an optimal path (state trajectory) of cost 5 (i.e., 5 parts is the minimum number of parts that must be moved to achieve an even distribution). The generalized Dijkstra's algorithm expands 36 states to find a solution. If we let $x_0 = [11\ 3\ 2]^t$ then $L = 5$ and $L_e = 1$. If we choose $X_f = \{[6\ 5\ 5]^t\}$, $A^*(\hat{h}_1(x_k))$ and $A^*(\hat{h}_\infty(x_k))$ both expand 11 states and result in a optimal state trajectory of cost 5. The generalized Dijkstra's algorithm expands 51 states to find a solution; hence, we see that for the 3-machine FMS, A^* using the heuristic functions specified via the results of Section IV far outperforms the generalized Dijkstra's algorithm.

For the 4-machine FMS in Fig. 1, let $\beta = 1$ and $x_1 = [0\ 5\ 2\ 6]^t$ so that $L = 3$ and $L_e = 1$. Choose $X_f = \{[4\ 3\ 3\ 3]^t, [3\ 3\ 3\ 4]^t\}$. $A^*(\hat{h}_1(x_k))$ and $A^*(\hat{h}_\infty(x_k))$ expand 38 and 53 states, respectively, and result in an optimal state trajectory of cost 6 that ends in $[3\ 3\ 3\ 4]^t$. The generalized Dijkstra's algorithm expands 141 states to find a solution to the SPP for the 4-machine FMS.

For the 6-machine FMS in Fig. 1, let $\beta = 1$ and $x_0 = [4\ 0\ 1\ 2\ 0\ 5]^t$ so that $L = 2$ and $L_e = 0$. Let $X_f = \{[2\ 2\ 2\ 2\ 2\ 2]^t\}$. $A^*(\hat{h}_1(x_k))$ expands 82 states and results in an optimal state trajectory of cost 6. The generalized Dijkstra's algorithm expanded 798 states to produce the same solution.

Note that if we had allowed $\beta > 1$ for the above examples then the computational savings obtained by using A^* over the generalized Dijkstra's algorithm would even be more pronounced. This is the case since A^* would exploit the fact that the robot could move multiple parts so that an even distribution could be achieved quicker. For the generalized Dijkstra's algorithm, large β will drastically increase the number of states it visits in finding an optimal state trajectory. Also note that for large N and total number of parts initially in the FMS, for many FMS track topologies the SPP can easily become too difficult to solve via any method due to combinatorial explosion. However, we have shown that for typical FMS systems the A^* algorithm, with the appropriate heuristic function, can solve the optimal parts distribution problem efficiently, and with significantly fewer computations than conventional techniques.

B. Artificial Intelligence Planning Problems

Several fundamental relationships between AI planning systems and control systems have recently been identified in [31]. Here we show that a class of AI planning problems falls into our framework

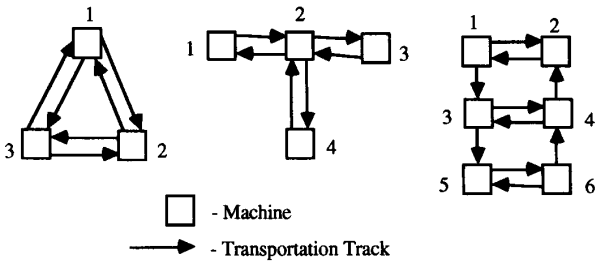

Fig. 1. Example flexible manufacturing system topologies.

and that the results of Section IV provide a method to specify heuristic functions so that SPP's can be solved efficiently for AI planning problems. The A^* algorithm has already been used for the solution to many AI planning problems such as tic-tac-toe, the 8 and 15 puzzle, etc. [34]. The extensions to the theory of heuristic search in this paper allow for a wider variety of such problems to be studied. For instance, in [27], the authors showed that the metric space approach could be used to specify the standard heuristic functions for the 8-puzzle, and discovered several new heuristics for this problem that also work for the more general N-puzzle. In [26] heuristic functions were specified for a "triangle and peg" problem and a simple robotics problem ("blocks world"). Here, we study the missionaries and cannibals Problem as in [26], an AI planning problem for which there currently exist no admissible and monotone heuristic functions (for any choice of the costs). In this way we illustrate that the results of Section IV facilitate the *discovery* of new heuristics.

The problem statement is as follows: Three missionaries and three cannibals are trying to cross a north–south river by crossing from east to west. As their only means of navigation, they have a small boat, which can hold one or two people. If the cannibals outnumber the missionaries on either side of the river, the missionaries will be eaten; this is to be avoided. Find a way to get them all across the river which minimizes the number of boat trips taken.

First, we model this problem with the model P. Let $X' = \mathbb{N}^6$ and $x_k = [x_1\ \ x_2\ \cdots\ x_6]^t$ and $x_{k+1} = [x_1'\ \ x_2'\ \cdots\ x_6']^t$ denote the current and next node (state), respectively. Let x_1 (x_4) and x_3 (x_6) denote the number of cannibals and missionaries on the east (west) side of the river, respectively. To model the part of the problem which states that "the cannibals cannot outnumber the missionaries" we let $X = X' - X_b$ where $X_b = \{x_k \in X: x_1 > x_3$ or $x_4 > x_6\}$. Let "E" and "W" denote the east and west side of the river, respectively. Let "C" and "M" denote cannibals and missionaries. Let $Q = \{q_i: i = 1, 2, \cdots, 10\}$ where $q_1 = 2\ C\ W \rightarrow E$ (move two cannibals from the west side of the river to the east side of the river); $q_2 = 2\ C\ E \rightarrow W$; $q_3 = 1\ C\ W \rightarrow E$; $q_4 = 1\ C\ E \rightarrow W$; $q_5 = 1\ C\ 1\ M\ W \rightarrow E$ (move one cannibal and one missionary from the west side of the river to the east side of the river); $q_6 = 1\ C\ 1\ M\ E \rightarrow W$; $q_7 = 1\ M\ E \rightarrow W$; $q_8 = 1\ M\ E \rightarrow W$; $q_9 = 2\ M\ E \rightarrow W$; $q_{10} = 2\ M\ W \rightarrow E$. Of course the boat moves in the indicated direction also. For the state transition function we have $\delta(q_2, [3\ 1\ 3\ 0\ 0\ 0]^t) = [1\ 0\ 3\ 2\ 1\ 0]^t$; the other cases are defined similarly. Let $\chi(x_k, x_{k+1}) = 1$ for all $(x_k, x_{k+1}) \in E(P)$, $x_0 = [3\ 1\ 3\ 0\ 0\ 0]^t$, and $X_f = \{[0\ 0\ 0\ 3\ 1\ 3]^t\}$. The SPP for the missionaries and cannibals problem is to find the minimum length sequence of inputs (loads of passengers) that will result in all persons on the west side of the river.

Currently, there does not exist any monotone $\hat{h}(x_k)$ for this problem. We now show that the results of Section IV allow for the specification of several such $\hat{h}(x)$. First consider ρ_p where $p = 2$ and notice that $\rho_2(x_k, x_{k+1}) \leq \sqrt{10}$ and $\chi(x_k, x_{k+1}) = 1$ for all $(x_k, x_{k+1}) \in E(P)$ so by Theorems 3 and 1 $\hat{h}(x_k) = (1/\sqrt{10})\rho_2(x_k, \bar{x})$ where $\bar{x} = [0\ 0\ 0\ 3\ 1\ 3]^t$ is an admissible and monotone heuristic function. Also notice that $\rho_\infty(x_k, x_{k+1}) \leq 2$ so by Theorems 3 and 1 $\hat{h}(x_k) = (1/2)\rho_\infty(x_k, \bar{x})$ where $\bar{x} = [0\ 0\ 0\ 3\ 1\ 3]^t$ is an admissible and monotone heuristic function. When these heuristic functions are used with A^* to find the solution to the SPP, the minimum length sequence of inputs found was: q_6, q_8, q_2, q_3, q_9, q_5, q_9, q_3, q_2, q_8, q_6. The solution involves 11 boat trips, the minimum number of trips needed to solve the problem.

VI. Conclusions

It was shown that for the class of problems modeled by P, Theorem 1 offers a method to specify admissible and monotone heuristic functions. In the case where $X \subset \mathbb{R}^n$ (e.g., for extended Petri nets), via Theorem 3 and Remark 2 we showed that our metric space approach can be used to *automatically* specify admissible and monotone heuristic functions. It was shown that if this heuristic function is subsequently used by A^*, it would, in a computationally efficient manner return a solution to a variety of SPP's for a wide class of applications. We showed via Theorems 4–6 that if the costs of the arcs could be modeled with a metric then further computational savings can be expected. We applied the results to an optimal parts distribution problem in flexible manufacturing systems and an AI planning problem. In each case, we showed that our main results in Section IV provided a technique to automatically specify an admissible and monotone $\hat{h}(x)$ and that when A^* uses this $\hat{h}(x)$ there is a significant reduction in the complexity of finding solutions to the SPP's.

References

[1] R. Bellman, *Dynamic Programming*, Princeton Univ. Press, NJ, 1957.

[2] R. Dechter, and J. Pearl, "Generalized best-first search strategies and the optimally of A^*," *Journal of the ACM*, vol. 32, no. 3, pp. 505–536, July 1985.

[3] J. Gaschnig, "A problem similarity approach to devising heuristics," *Proc. 6th IJCAI*, Tokyo, Japan, Aug. 1977, pp. 301–307.

[4] D. Gelperin, "On the Optimality of A^*," *Art. Intell.* vol. 8, pp. 69–76, 1977.

[5] B. L. Golden, and M. Ball, "Shortest paths with Euclidean distances: An Explanatory Model," *Networks*, vol. 8, pp. 297–314, 1978.

[6] M. Gondran, and M. Minoux, *Graphs and Algorithms*, Wiley, NY, 1984.

[7] G. Guida, and M. Somalvico, "Semantics in problem representation and search," *Inf. Proc. Letters*, vol. 5, no. 5, pp. 141–145, 1976.

[8] G. Guida, and M. Somalvica, "A method for computing heuristics in problem solving," *Information Sciences*, vol. 19, pp. 251–259, 1979.

[9] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. on Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, July 1968.

[10] P. E. Hart, N. J. Nilsson, and B. Raphael, "Correction to: a formal basis for the heuristic determination of minimum cost paths," *SIGART Newsletter*, vol. 37, pp. 28–29, 1972.

[11] M. Held, and R. M. Karp, "The traveling salesman problem and minimum spanning trees," *Operations Research*, vol. 18, pp. 1138–1162, 1970.

[12] L. E. Holloway, and B. H. Krogh, "Synthesis of feedback control logic for a class of controlled Petri nets," *IEEE Trans. on Automatic Control*, vol. 35, no. 5, pp. 514–523, May 1990.

[13] T. Ibaraki, "Branch and bound procedure and state space representation of combinatorial optimization problems," *Information and Control*, vol. 36, no. 1, pp. 1–27, Jan. 1978.

[14] K. B. Irani, and S. I. Yoo, "A methodology for solving problems: problem modeling and heuristic generation," *IEEE Trans. on Pattern Anal. and Mach. Int.*, vol. 10, no. 5, pp. 676–686, Sept. 1988.

[15] R. M. Karp, and M. Held, "Finite-state processes and dynamic programming," *SIAM J. Applied Math*, vol. 15, no. 3, pp. 693–718, May 1967.

[16] B. H. Krogh, "Controlled petri nets and maximally permissive feedback logic," *Proc. of the Allerton Conf. on Communication, Control, and Computing*, Univ. of Illinois, pp. 317–326, Oct. 1987.

[17] E. L. Lawler, and D. E. Wood, "Branch and bound methods: A survey," *Op. Res.*, vol. 14, no. 4, pp. 699–719, July–Aug. 1966.

[18] Y. Li, and W. M. Wonham, "A state-variable approach to the modeling and control of discrete event systems," *Proc. of the 26th Allerton Conf. on Communication, Control, and Computing*, Univ. of Illinois, Champaign-Urbana, Sept. 1988, pp. 1140–1149.

[19] Y. Li, and W. M. Wonham, "A^* algorithm for vector discrete event systems, *Systems Control Group Technical Note 891006*, Oct. 1989.

[20] A. Martelli, "On the search complexity of admissible search algorithms," *AI*, vol. 8, pp. 1–13, 1977.

[21] A. N. Michel, and C. J. Herget, *Mathematical Foundations in Engineering and Science: Algebra and Analysis*, Prentice-Hall, NJ, 1981.

[22] T. L. Morin, and T. L. Marsten, "Branch and Bound Strategies for Dynamic Programming," *Operations Res.*, vol. 24, no. 4, pp. 611–627, July–Aug. 1976.

[23] D. S. Nau, V. Kumar, and L. Kanal, "General branch and bound and its relation to A* and AO*, *Art. Intell.*, vol. 23, pp. 29–58, 1984.

[24] N. J. Nilsson, *Problem-Solving Methods in Artificial Intelligence*, McGraw-Hill, NY, 1971.

[25] N. J. Nilsson, *Principles of Artificial Intelligence*, Tioga, NY, 1980.

[26] K. M. Passino, and P. J. Antsaklis, "Artificial intelligence planning problems in a petri net framework," *Proc. of the American Control Conf.*, pp. 626–631, Atlanta, GA, June 1988.

[27] K. M. Passino, and P. J. Antsaklis, "Planning via heuristic search in a petri net framework," *Proc. of the 3rd IEEE Int. Symp. on Intelligent Control*, Arlington, VA, August 1988, pp. 350–355.

[28] K. M. Passino, *Analysis and Synthesis of Discrete Event Regulator Systems*, Ph.D. Dissertation, Dept. of Elec. and Comp. Eng., Univ. Notre Dame, April 1989.

[29] K. M. Passino, and P. J. Antsaklis, "Near optimal control of discrete event systems," *Proc. of the Allerton Conf. Communication, Control, and Computing*, Univ. of Illinois, Sept. 1989, pp. 915–924.

[30] K. M. Passino, and P. J. Antsaklis, "On the optimal control of discrete event systems," *Proc. of the Conf. Decision and Control*, Tampa, Florida, Dec. 1989, pp. 2713–2718.

[31] K. M. Passino, and P. J. Antsaklis, "A system and control theoretic perspective on artificial intelligence planning systems," *Int. Journal Appl. Art. Intell.*, vol. 3, no. 1, pp. 1–32, 1989.

[32] K. M. Passino, and P. J. Antsaklis, "Optimal stabilization of discrete event systems," *Proc. of the IEEE Conf. on Dec. and Control*, Hawaii, Dec. 1990, pp. 670–671.

[33] J. Pearl, "On the discovery and generation of certain heuristics," *The AI Mag.*, vol. 4, no. 1., pp. 23–33, 1983.

[34] J. Pearl, *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley, Reading, MA, 1984.

[35] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, NJ, 1981.

[36] P. J. Ramadge, and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control Optimization*, vol. 25, no. 1, pp. 206–230, Jan. 1987.

[37] R. Sedgewick, and J. S. Vitter, "Shortest paths in euclidean graphs," *Algorithms*, vol. 1, pp. 31–48, 1986.

[38] T. Ushio, "Maximally permissive feedback and modular control synthesis in petri nets with external input places," *IEEE Trans. on Automatic Control*, vol. 35, no. 7, pp. 844–848, July 1990.

[39] M. Valtorta, "A result on the computational complexity of heuristic estimates for the A^* algorithm," *Information Sciences*, vol. 34, pp. 47–59, 1984.

[40] D. J. White, *Dynamic Programming*, Holden-Day, San Francisco, CA, 1969.